

# Contents

Foreword . . . . .	xi
Introduction . . . . .	xiii
1. Global Setup . . . . .	1
1.1 Setup	1
1.2 The Examples	1
1.3 C++ Compiler	3
1.4 CMake	6
1.5 Google Mock	6
1.6 CppUTest	8
1.7 libcurl	9
1.8 JsonCpp	10
1.9 rlog	10
1.10 Boost	12
1.11 Building Examples and Running Tests	12
1.12 Teardown	13
2. Test-Driven Development: A First Example . . . . .	15
2.1 Setup	15
2.2 The Soundex Class	15
2.3 Getting Started	16
2.4 Fixing Unclean Code	23
2.5 Incrementalism	25
2.6 Fixtures and Setup	28
2.7 Thinking and TDD	30
2.8 Test-Driving vs. Testing	33
2.9 What If?	36
2.10 One Thing at a Time	37
2.11 Limiting Length	39
2.12 Dropping Vowels	40

2.13	Doing What It Takes to Clarify Tests	41
2.14	Testing Outside the Box	43
2.15	Back on Track	45
2.16	Refactoring to Single-Responsibility Functions	46
2.17	Finishing Up	48
2.18	What Tests Are We Missing?	48
2.19	Our Solution	49
2.20	The Soundex Class	50
2.21	Teardown	54
<b>3.</b>	<b>Test-Driven Development Foundations . . . . .</b>	<b>55</b>
3.1	Setup	55
3.2	Unit Test and TDD Fundamentals	55
3.3	The TDD Cycle: Red-Green-Refactor	57
3.4	The Three Rules of TDD	59
3.5	Getting Green on Red	60
3.6	Mind-Sets for Successful Adoption of TDD	69
3.7	Mechanics for Success	73
3.8	Teardown	77
<b>4.</b>	<b>Test Construction . . . . .</b>	<b>79</b>
4.1	Setup	79
4.2	Organization	79
4.3	Fast Tests, Slow Tests, Filters, and Suites	86
4.4	Assertions	89
4.5	Inspecting Privates	96
4.6	Testing vs. Test-Driving: Parameterized Tests and Other Toys	100
4.7	Teardown	103
<b>5.</b>	<b>Test Doubles . . . . .</b>	<b>105</b>
5.1	Setup	105
5.2	Dependency Challenges	105
5.3	Test Doubles	106
5.4	A Hand-Crafted Test Double	107
5.5	Improving Test Abstraction When Using Test Doubles	112
5.6	Using Mock Tools	114
5.7	Getting Test Doubles in Place	123
5.8	Design Will Change	130
5.9	Strategies for Using Test Doubles	132

5.10	Miscellaneous Test Double Topics	136
5.11	Teardown	138
<b>6.</b>	<b>Incremental Design . . . . .</b>	<b>141</b>
6.1	Setup	141
6.2	Simple Design	141
6.3	Where Is the Up-Front Design?	166
6.4	Refactoring Inhibitors	169
6.5	Teardown	171
<b>7.</b>	<b>Quality Tests . . . . .</b>	<b>173</b>
7.1	Setup	173
7.2	Tests Come FIRST	173
7.3	One Assert per Test	178
7.4	Test Abstraction	181
7.5	Teardown	194
<b>8.</b>	<b>Legacy Challenges . . . . .</b>	<b>195</b>
8.1	Setup	195
8.2	Legacy Code	195
8.3	Themes	196
8.4	The Legacy Application	198
8.5	A Test-Driven Mentality	201
8.6	Safe Refactoring to Support Testing	202
8.7	Adding Tests to Characterize Existing Behavior	205
8.8	Sidetracked by the Reality of Legacy Code	206
8.9	Creating a Test Double for rlog	207
8.10	Test-Driving Changes	211
8.11	A New Story	213
8.12	A Brief Exploration in Seeking Faster Tests	214
8.13	Mondo Extracto	215
8.14	Spying to Sense Using a Member Variable	218
8.15	Spying to Sense Using a Mock	219
8.16	Alternate Injection Techniques	224
8.17	Large-Scale Change with the Mikado Method	224
8.18	An Overview of the Mikado Method	225
8.19	Moving a Method via Mikado	226
8.20	More Thoughts on the Mikado Method	236
8.21	Is It Worth It?	237
8.22	Teardown	238



<b>9.</b>	<b>TDD and Threading . . . . .</b>	<b>239</b>
9.1	Setup	239
9.2	Core Concepts for Test-Driving Threads	239
9.3	The GeoServer	240
9.4	Performance Requirements	246
9.5	Designing an Asynchronous Solution	249
9.6	Still Simply Test-Driving	252
9.7	Ready for a Thready!	254
9.8	Exposing Concurrency Issues	256
9.9	Creating Client Threads in the Test	259
9.10	Creating Multiple Threads in the ThreadPool	261
9.11	Back to the GeoServer	263
9.12	Teardown	267
<b>10.</b>	<b>Additional TDD Concepts and Discussions . . . . .</b>	<b>269</b>
10.1	Setup	269
10.2	TDD and Performance	269
10.3	Unit Tests, Integration Tests, and Acceptance Tests	278
10.4	The Transformation Priority Premise	281
10.5	Writing Assertions First	294
10.6	Teardown	298
<b>11.</b>	<b>Growing and Sustaining TDD . . . . .</b>	<b>299</b>
11.1	Setup	299
11.2	Explaining TDD to Nontechies	300
11.3	The Bad Test Death Spiral, aka the SCUMmy Cycle	304
11.4	Pair Programming	306
11.5	Katas and Dojos	310
11.6	Using the Code Coverage Metric Effectively	313
11.7	Continuous Integration	314
11.8	Deriving Team Standards for TDD	315
11.9	Keeping Up with the Community	316
11.10	Teardown	317
<b>A1.</b>	<b>Comparing Unit Testing Tools . . . . .</b>	<b>319</b>
A1.1	Setup	319
A1.2	TDD Unit Testing Tool Features	319
A1.3	Notes on Google Mock	321
A1.4	Notes on CppUTest	321
A1.5	Other Unit Testing Frameworks	321
A1.6	Teardown	322

<b>A2. Code Kata: Roman Numeral Converter</b>	<b>323</b>
A2.1 Setup	323
A2.2 Let's Go!	323
A2.3 Practice Makes Perfect	331
A2.4 Teardown	331
<b>A3. Bibliography</b>	<b>333</b>
Index	335